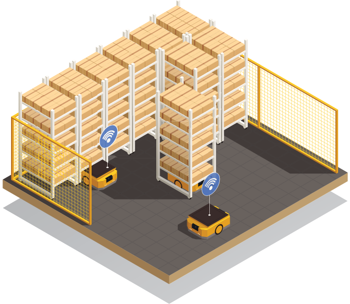
Multiagent Rollout with Reshuffling for Warehouse Robots Path Planning

William Emanuelsson, Alejandro Penacho Riveiros, Yuchao Li, Karl H. Johansson, Jonas Mårtensson

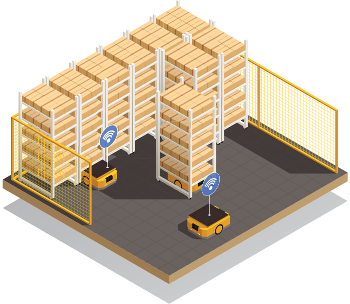
Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm, Sweden

Introduction: Warehouse Problem



- We consider the warehouse robots path finding problem.

Introduction: Warehouse Problem



- We consider the warehouse robots path finding problem.
- The goal is to compute 'optimal' paths for all robots to their respective targets while avoiding collisions.

Introduction: Warehouse Problem

- We consider the warehouse robots path finding problem.
- The goal is to compute 'optimal' paths for all robots to their respective targets while avoiding collisions.
- The problem is often modeled as computing shortest paths in a grid world environment: discrete optimal control problem.

Introduction: Existing Approaches and Our Method

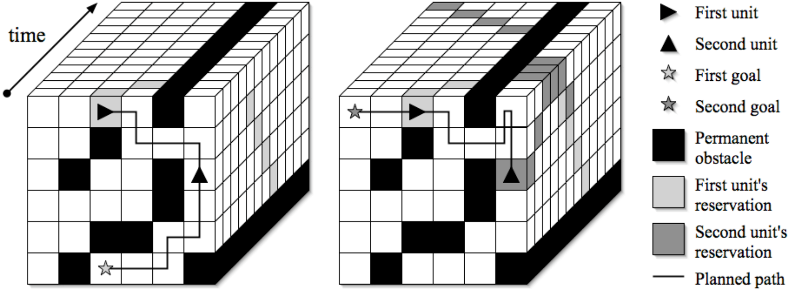


Figure source: [Sil05, Fig. 1].

- Existing approaches, such as [Sil05], create a space-time grid environment and apply shortest path algorithms for agents one at a time.

Introduction: Existing Approaches and Our Method

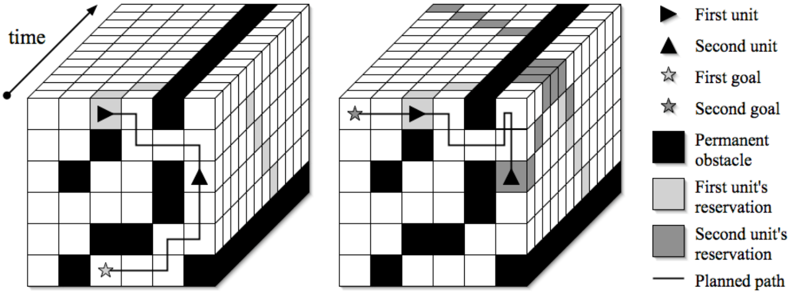


Figure source: [Sil05, Fig. 1].

- Existing approaches, such as [Sil05], create a space-time grid environment and apply shortest path algorithms for agents one at a time.
- By treating preceding robots as dynamic obstacles, collisions are avoided.

Introduction: Existing Approaches and Our Method

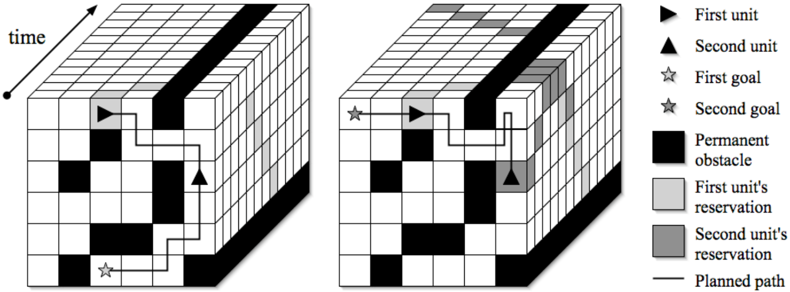


Figure source: [Sil05, Fig. 1].

- Existing approaches, such as [Sil05], create a space-time grid environment and apply shortest path algorithms for agents one at a time.
- By treating preceding robots as dynamic obstacles, collisions are avoided.
- Our approach uses the static space grid and relies on simulation to detect collisions between robots.

Dynamic Programming Model

- **State space** is denoted as X; the state at kth stage as x_k; the **control constraint set** at x_k as $U(x_k)$. **System dynamics** and **stage costs** are denoted as f and g respectively.

Dynamic Programming Model

- **State space** is denoted as X; the state at kth stage as x_k; the **control constraint set** at x_k as $U(x_k)$. **System dynamics** and **stage costs** are denoted as f and g respectively.
- For a policy $\mu : X \mapsto U$ such that control constraints are respected $\mu(x) \in U(x)$, its **cost function** is defined as

$$J_\mu(x_0) = \sum_{k=0}^{\infty} \alpha^k g(x_k, \mu(x_k)),$$

where $\alpha \in (0, 1)$ is the discount factor.

Dynamic Programming Model

- **State space** is denoted as X; the state at kth stage as x_k; the **control constraint set** at x_k as $U(x_k)$. **System dynamics** and **stage costs** are denoted as f and g respectively.
- For a policy $\mu : X \mapsto U$ such that control constraints are respected $\mu(x) \in U(x)$, its **cost function** is defined as
$$J_\mu(x_0) = \sum_{k=0}^{\infty} \alpha^k g(x_k, \mu(x_k)),$$
where $\alpha \in (0, 1)$ is the discount factor.
- The goal is to obtain optimal policy μ^* such that $J_{\mu^*}(x) = J^*(x)$ for all x, where J^* is defined as
$$J^*(x_0) = \inf_{\substack{u_k \in U(x_k),\ k=0,1,\ldots \\ x_{k+1}=f(x_k,u_k),\ k=0,1,\ldots}} \sum_{k=0}^{\infty} \alpha^k g(x_k, u_k).$$

Multiagent Problem with Classical Information Pattern

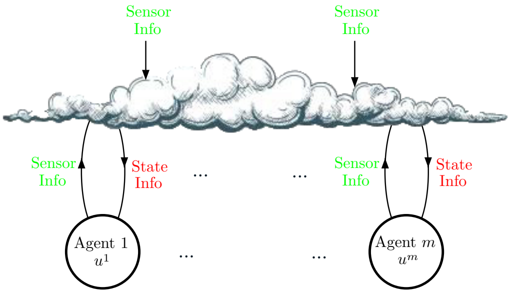


Figure source: [Ber21, Fig. 1]

- Discrete optimal control problem: both state and control spaces X and U are finite.

Multiagent Problem with Classical Information Pattern

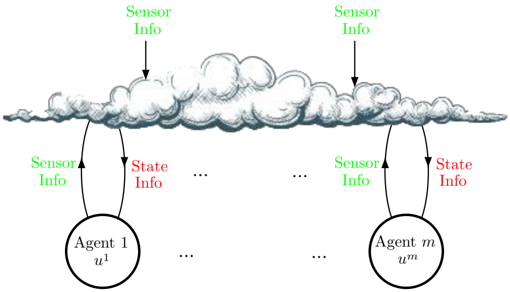


Figure source: [Ber21, Fig. 1]

- Discrete optimal control problem: both state and control spaces X and U are finite.
- Decision/control has m components $u = (u^1, \ldots, u^m)$ corresponding to m 'agents.'

Multiagent Problem with Classical Information Pattern

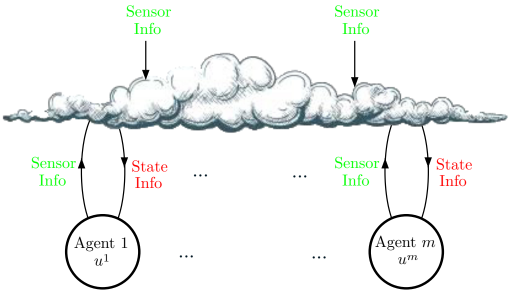


Figure source: [Ber21, Fig. 1]

- Discrete optimal control problem: both state and control spaces X and U are finite.
- Decision/control has m components $u = (u^1, \dots, u^m)$ corresponding to m 'agents.'
- The agents operate as a team to minimize the shared costs.

Rollout Method [TG96]

- If we can compute the function J^* offline, the optimal policy μ^* can be obtained via

$$\mu^*(x) \in \arg\min_{u \in U(x)} \Big\{ g(x,u) + \alpha J^*\big(f(x,u)\big) \Big\}.$$

Rollout Method [TG96]

- If we can compute the function J^* offline, the optimal policy μ^* can be obtained via

$$\mu^*(x) \in \arg \min_{u \in U(x)} \Big\{ g(x,u) + \alpha J^*\big(f(x,u)\big) \Big\}.$$

- Due to large size of the state space X, the function J^* can not be obtained.

Rollout Method [TG96]

- If we can compute the function J^* offline, the optimal policy μ^* can be obtained via

$$\mu^*(x) \in \arg \min_{u \in U(x)} \Big\{ g(x,u) + \alpha J^*\big(f(x,u)\big) \Big\}.$$

- Due to large size of the state space X, the function J^* can not be obtained.
- Rollout method relies on a known policy μ, called **base policy**, and apply to the system the **rollout policy** $\tilde{\mu}$ defined as

$$\tilde{\mu}(x) \in \arg \min_{u \in U(x)} \Big\{ g(x,u) + \alpha J_\mu\big(f(x,u)\big) \Big\},$$

where the related values of J_μ need to be known.

Rollout Method [TG96]

- If we can compute the function J^* offline, the optimal policy μ^* can be obtained via

$$\mu^*(x) \in \arg\min_{u \in U(x)} \Big\{ g(x,u) + \alpha J^*\big(f(x,u)\big) \Big\}.$$

- Due to large size of the state space X, the function J^* can not be obtained.
- Rollout method relies on a known policy μ, called **base policy**, and apply to the system the **rollout policy** $\tilde{\mu}$ defined as

$$\tilde{\mu}(x) \in \arg\min_{u \in U(x)} \Big\{ g(x,u) + \alpha J_\mu\big(f(x,u)\big) \Big\},$$

where the related values of J_μ need to be known.
- Cost improvement property [BTW97]:

$$J^*(x) \le J_{\tilde{\mu}}(x) \le J_\mu(x), \quad \forall x \in X.$$

Challenges in Rollout for Multiagent Problem

For the multiagent problem with $U(x) = \big(U^1(x), \ldots, U^m(x)\big)$, rollout takes the form:

$$\big(\tilde{\mu}^1(x), \ldots, \tilde{\mu}^m(x)\big) \in \arg \min_{(u^1,\ldots,u^m)\in U(x)} \Big\{ g(x, u^1, \ldots, u^m) + \alpha J_\mu\big(f(x, u^1, \ldots, u^m)\big) \Big\}$$

For this method to be applied to the problem, there are two major challenges.

- Suppose $|U^i(x)| \leq C$. The search space grow exponentially with m.

Challenges in Rollout for Multiagent Problem

For the multiagent problem with $U(x) = \big(U^1(x), \ldots, U^m(x)\big)$, rollout takes the form:

$$\big(\tilde{\mu}^1(x), \ldots, \tilde{\mu}^m(x)\big) \in \arg \min_{(u^1,\ldots,u^m)\in U(x)} \Big\{ g(x, u^1, \ldots, u^m) + \alpha J_\mu\big(f(x, u^1, \ldots, u^m)\big) \Big\}$$

For this method to be applied to the problem, there are two major challenges.

- Suppose $|U^i(x)| \leq C$. The search space grow exponentially with m.
- The values of J_μ cannot be computed beforehand.

Multiagent Rollout [Ber21]

- Multiagent rollout performs m minimization in succession, one-agent-at-a-time

$$\tilde{\mu}^1(x) \in \arg\min_{u^1} \Big\{ g(x, u^1, \mu^2(x) \ldots, \mu^m(x)) + \alpha J_\mu\big(f(x, u^1, \mu^2(x) \ldots, \mu^m(x))\big) \Big\}$$

$$\tilde{\mu}^2(x) \in \arg\min_{u^2} \Big\{ g(x, \tilde{\mu}^1(x), u^2, \mu^3(x) \ldots, \mu^m(x)) + \alpha J_\mu\big(f(x, \tilde{\mu}^1(x), u^2, \mu^3(x) \ldots, \mu^m(x))\big) \Big\}$$

$$\ldots \quad \ldots \quad \ldots$$

$$\tilde{\mu}^m(x) \in \arg\min_{u^m} \Big\{ g(x, \tilde{\mu}^1(x), \ldots, \tilde{\mu}^{m-1}(x), u^m) + \alpha J_\mu\big(f(x, \tilde{\mu}^1(x), \ldots, \tilde{\mu}^{m-1}(x), u^m)\big) \Big\}$$

the search space grow linearly with m!

Multiagent Rollout [Ber21]

- Multiagent rollout performs m minimization in succession, one-agent-at-a-time

$$\tilde{\mu}^1(x) \in \arg\min_{u^1} \Big\{ g(x, u^1, \mu^2(x) \dots, \mu^m(x)) + \alpha J_\mu\big(f(x, u^1, \mu^2(x) \dots, \mu^m(x))\big) \Big\}$$

$$\tilde{\mu}^2(x) \in \arg\min_{u^2} \Big\{ g(x, \tilde{\mu}^1(x), u^2, \mu^3(x) \dots, \mu^m(x)) + \alpha J_\mu\big(f(x, \tilde{\mu}^1(x), u^2, \mu^3(x) \dots, \mu^m(x))\big) \Big\}$$

$$\dots \quad \dots \quad \dots$$

$$\tilde{\mu}^m(x) \in \arg\min_{u^m} \Big\{ g(x, \tilde{\mu}^1(x), \dots, \tilde{\mu}^{m-1}(x), u^m) + \alpha J_\mu\big(f(x, \tilde{\mu}^1(x), \dots, \tilde{\mu}^{m-1}(x), u^m)\big) \Big\}$$

  the search space grow linearly with m!
- The values of J_μ can be computed as needed via real-time simulation!

Multiagent Rollout with Reshuffle

- What if the obtained control $\tilde{\mu}(x)$ is not 'good' enough?

Multiagent Rollout with Reshuffle

- What if the obtained control $\tilde{\mu}(x)$ is not 'good' enough?
- Instead of using a fixed agent order, randomly generate a permutation function $\sigma : \{1, \ldots, m\} \mapsto \{1, \ldots, m\}$! The controls of agents are computed according to the new order defined by σ.

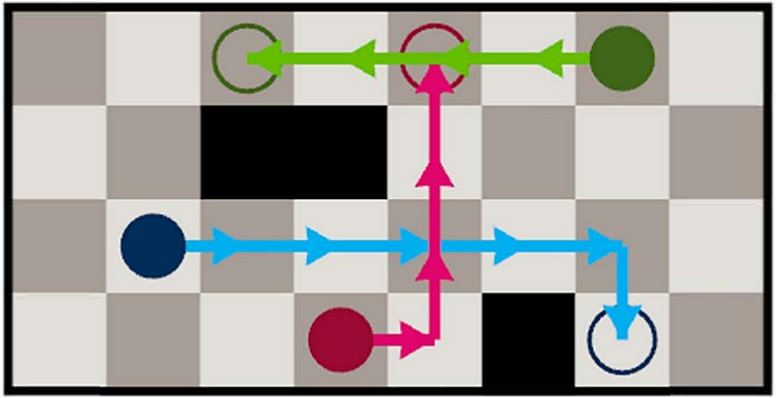
Multiagent Rollout with Reshuffle

- What if the obtained control $\tilde{\mu}(x)$ is not 'good' enough?
- Instead of using a fixed agent order, randomly generate a permutation function $\sigma : \{1, \ldots, m\} \mapsto \{1, \ldots, m\}$! The controls of agents are computed according to the new order defined by σ.
- Proposition 8.1 (informal): For the obtained policy $\tilde{\mu}$, we have that

$$J_{\tilde{\mu}}(x) \leq \tilde{J}(x) \leq J_{\mu}(x),$$

for all x, where $\tilde{J}(x) = g\big(x, \tilde{\mu}(x)\big) + \alpha J_{\mu}\Big(f\big(x, \tilde{\mu}(x)\big)\Big)$.

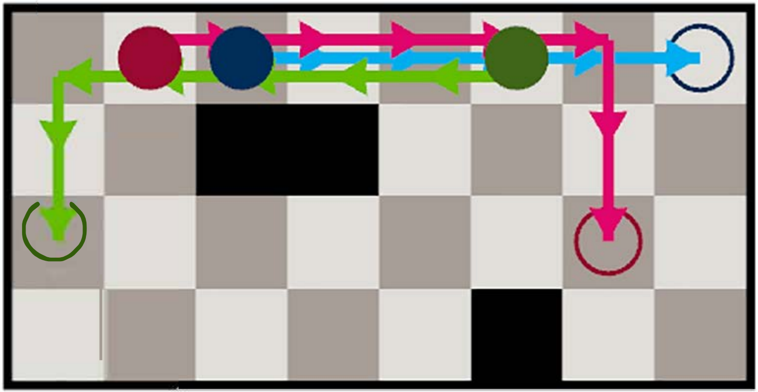
Application to the Warehouse Problem



3 agents move in 4 directions with perfect vision. They have been assigned to some targets. Objective is to reach their respective targets in minimum time while avoiding collision.[a]

[a]Video credit: Alejandro Penacho Riveiros

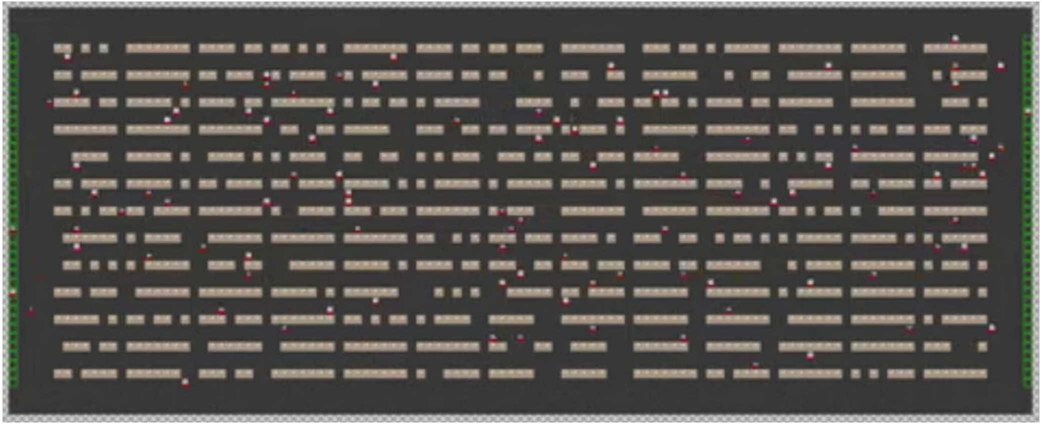
Application to the Warehouse Problem



3 agents move in 4 directions with perfect vision. They have been assigned to some targets. Objective is to reach their respective targets in minimum time while avoiding collision.[a]

[a] Video credit: Alejandro Penacho Riveiros

Application to the Warehouse Problem



200 agents move in 4 directions with perfect vision. They have been assigned to some targets. Objective is to deliver 1183 goods in minimum time while avoiding collision.[a]

[a] See `https://github.com/will-em/multi-agent-rollout` for implementation.

Conclusions

- We introduced the multiagent rollout with reshuffling algorithm;

Conclusions

- We introduced the multiagent rollout with reshuffling algorithm;
- The scheme was applied to large-scale warehouse robots path planning problem;

Conclusions

- We introduced the multiagent rollout with reshuffling algorithm;
- The scheme was applied to large-scale warehouse robots path planning problem;
- Through online replanning, the method can adapt to a changing environment.

References I

[Ber21] Dimitri P. Bertsekas. Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/CAA Journal of Automatica Sinica*, 8(2):249–272, 2021.

[BTW97] Dimitri P. Bertsekas, John N. Tsitsiklis, and Cynara Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3(3):245–262, 1997.

[Sil05] David Silver. Cooperative pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 1, pages 117–122, 2005.

[TG96] Gerald Tesauro and Gregory Galperin. On-line policy improvement using Monte-Carlo search. *Advances in Neural Information Processing Systems*, 9, 1996.